Information Sciences 647 (2023) 119463

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Taming over-smoothing representation on heterophilic graphs

Kai Guo [a,c], Xiaofeng Cao [a,c,*], Zhining Liu [b], Yi Chang [a,c,*]

[a] School of Artificial Intelligence, Jilin University, Changchun 130012, China
[b] Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, United States of America
[c] Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, Ministry of Education, China

ARTICLE INFO

Keywords:
Graph neural networks
Heterophilic graphs
Over-smoothing
Label estimation-based message passing

ABSTRACT

Graph Neural Networks (GNNs) have shown expressive modeling ability on graphs. An inherent assumption is that those connected graph nodes exhibit strong homophily, passing consistent label information into the associated central node representation. However, heterophilic graphs may hold inconsistent labels for those connected nodes, degenerating the performance of the typical message passing and exacerbating its over-smoothing issue. In this paper, to tame heterophilic graphs for GNNs, a novel label estimation-based message passing scheme is presented, which can further relieve the over-smoothing and lead the learned representations to be more distinguishable on graphs with either homophily or heterophily. Specifically, we optimize the edge weights of the graph by invoking the label estimation with the goal of returning a more effective adjacency matrix. With this matrix, a novel message passing scheme is presented, aggregating neighbor information with consistent labels to their central nodes. Meanwhile, a symmetric cross entropy is also employed to mitigate negative effects from noisy labels. With such guarantees in our new message passing, a uniform framework termed Label Estimation-based GNN (LE-GNN) is finally established. Extensive experiments demonstrate the ability of our model to prevent over-smoothing and show its effectiveness for node classification on both homophilic and heterophilic graphs.

1. Introduction

Graph neural networks (GNNs) have achieved impressive performance in learning representation of social networks [1,2], molecular networks [3,4] and citation networks [5,6], etc. The representations of those graph-structured data provide effective embeddings for associated downstream tasks such as node classification [7,5,8], graph classification [9–11] and drug discovery [12,13]. To learn expressive embeddings, two operations in GNN: feature transformation and neighbor aggregation, are employed to iteratively update node representation from neighbors, where the neighbor aggregation is typically implemented using message passing.

GNNs conduct message passing along the adjacency edges based on the assumption that networks with strong homophily, where linked nodes have similar features and belong to the same class. For example, friends tend to have similar hobbies. This traditional aggregation method is mainly achieved by operations such as sum and mean, which is equivalent to a kind of adjacency smoothing process, making the features of adjacent nodes more similar [14]. Thus, traditional message passing mechanism performs well in graph with homophily. However, the principle of "opposites attract" [15] also exists in the real world, leading to networks with heterophily: linked nodes tend to have inconsistent labels, as shown in Fig. 1. Different types of amino acids, for instance, are more

* Corresponding authors.
E-mail addresses: xiaofengcao@jlu.edu.cn (X. Cao), yichang@jlu.edu.cn (Y. Chang).

https://doi.org/10.1016/j.ins.2023.119463
Received 18 September 2022; Received in revised form 15 November 2022; Accepted 5 August 2023

Available online 10 August 2023

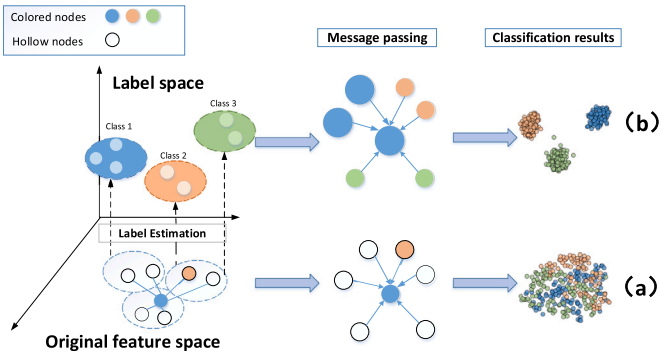

Fig. 1. In the heterophilic graphs, the neighbors of central nodes may have inconsistent labels, where colored nodes are annotated with different labels, hollow nodes are unannotated. **(a)** Traditional message passing is disastrous for capturing heterophily. It leads nodes with different labels to be mixed together, resulting in learning indistinguishable node representations. **(b)** Our proposed message passing mechanism introduces trustworthy label estimation to further clarify the classification boundaries.

likely to be linked in protein structure [15]. Therefore, the traditional message passing mechanism performs well in graphs with homophily while failing to generalize to heterophilic graphs.

The reason is that this traditional message passing mechanism delivers low-quality neighbor information, which is disastrous for node representation of heterophilic graphs. It causes nodes with different labels to be mixed together, making the model unable to learn the distinguished node representation. This low-quality information produces mistakes for downstream tasks, such as node classification, etc. Meanwhile, with the number of layers increases, the mistakes become more serious, then leads to the exacerbation of the over-smoothing, difference degeneration among nodes of different classes, and inability of node classification [16], as shown in Fig. 1.(a).

Thus, despite GNN and their variants [17,16,18] having achieved significant success in graphs with homophily, the heterophilic graph challenges existing GNNs to obtain discriminative representations. Recently, several studies have been concerned with this challenge, H2GNN [15] identified a key design—ego- and neighbor-embedding separations—that boosts learning from the graph structure under heterophily. However, it still suffers from over-smoothing issues, where learned node embeddings have quite similar representations [19]. Geom-GCN [20] proposes a geometric aggregation approach to exploit the structural information of nodes in the neighborhood and to capture an extensive range of dependencies of nodes in the heterophilic graphs. GCNII [21] captures heterophily by preventing the problem of over-smoothing, which suggests a relationship between over-smoothing and heterophily.

In this work, we hold that label-guide message passing can deliver reliable information for learning graph representation. The closer the labels are to the correct ones, the more discriminative the learned node representations are. Therefore, we face the following challenges: (1) How to use the incomplete labels to guide the message passing mechanism to spread reliable information? (2) If there are errors in label information, how to reduce the influence of noisy labels and make the model robust?

The above challenges motivate us to propose a label estimation-based message passing for node classification both on graphs with homophily and heterophily. We unify features, topology, and labels in the GNN model by additionally considering label space. Specifically, we use an effective label estimation to map the node attributes of feature space into their associated label space. Then, we integrate the estimated labels with given ground-truth labels of the training set. Based on these labels, we obtain a more precise adjacency matrix in which edges with consistent labels are given larger weights. Additionally, we employ symmetric cross entropy loss to enhance robustness of our model to noisy labels. Therefore, as shown in Fig. 1.(b), more information about nodes with the same label would be aggregated along the edges, which further clarifies the classification boundaries. With such improvements, the representations on graphs with either homophily or heterophily become more distinguishable, even when stacking more layers. Contributions are summarized as following.

- We propose a novel label estimation-based GNN framework to facilitate node classification on homophilic and heterophilic graphs.
- Our proposed message passing mechanism tames over-smoothing representation by introducing trustworthy label estimation, further clarifying the classification boundaries.
- We conduct comprehensive experiments on the node classification tasks to demonstrate the general effectiveness of our proposed model.

2. Related work

GNNs. Several studies have defined the concept of convolution on graphs, inspired by the tremendous success of CNNs. GNNs have achieved better performance for various real-world problems such as node classification and graph classification by learning graph representation. GCN is the first notable research on GNNs, which advances graph convolution based on spectral graph theory [14]. Later, Veličković et al. [17], Li et al. [22] performed improvements, extensions and approximations of spectral-based GCNs. Specifically, GCN considers node features and graph connectivity structure together and iteratively aggregates neighbor represen-

Table 1
Summary of the main symbols and notations.

Symbols and Notations	
Symbol	Description
$\boldsymbol{X} \in \mathbb{R}^{n\times d}$	Input feature matrix
$\boldsymbol{A} \in \mathbb{R}^{n\times n}$	Adjacency matrix of graph
$\boldsymbol{D} \in \mathbb{R}^{n\times n}$	Diagonal degree matrix
$\boldsymbol{I} \in \mathbb{R}^{n\times n}$	Identity matrix
$\hat{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$	Adjacency matrix with self-loop
$\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$	Normalized adjacency matrix with self-loop
$\boldsymbol{L} \in \mathbb{R}^{n\times n}$	Graph laplacian matrix
$\boldsymbol{H}^l \in \mathbb{R}^{n\times d}$	Embedding matrix in layer l
$\boldsymbol{W}^l \in \mathbb{R}^{d_f\times d}$	Weight matrix in layer l
$\boldsymbol{Y} \in \mathbb{R}^{n\times c}$	Ground-truth label matrix
$\tilde{\boldsymbol{Y}}^E \in \mathbb{R}^{n\times c}$	Estimated label matrix
$\boldsymbol{S} \in \mathbb{R}^{n\times n}$	Similarity matrix
$\mathbb{E}$	Cross entropy
σ	Activation function
α	Parameter used to tune the trade-off between the current layer and the input layer
λ_1	Trade-off parameter used to tune the importance of cross entropy loss
λ_2	Trade-off parameter used to tune the importance of symmetric cross entropy loss

tations to the central node by applying both feature transformation and neighbor aggregation operations. Besides, GAT utilizes the attention mechanism to select the essential nodes by learning the different weights for neighbor nodes [17]. However, these models achieve their best performance with only two layers, which only consider immediate neighbors, and the performance degrades greatly when stacking more layers. Some studies attribute this phenomenon to the over-smoothing problem, where node representations are similar and indistinguishable after stacking multiple layers. Several models have recently been developed to address the over-smoothing problem such as APPNP [16], JKNet [23], DropEdge [24] and GCNII [21]. For example, APPNP alleviates over-smoothing issue by using PageRank-style propagation. To avoid over-smoothing, GCNII obtains long-range dependences by using initial residual connection and identity mapping.

Heterophily Most prior work considers graphs with strong homophily. However, there are graphs with strong heterophily in the real world, where previous work fails to perform well. Therefore, several studies transfer their attendance to explore to address the problem of heterophilic graphs. For example, Geom-GCN proposes a novel geometric aggregation scheme for graph neural networks, in which aggregation on a graph can benefit from a continuous space underlying the graph. In this way, Geom-GCN [20] achieves improvement over GCN by a great margin on the heterophilic graphs. H2GNN [15] proposes a set of fundamental designs, such as higher-order neighborhoods, ego- and neighbor-embedding separation, and a combination of intermediate representations, that enhance learning from graph structure under heterophily. TDGNN [25] presents a tree decomposition method for disentangling neighborhoods in distinct layers in order to reduce feature smoothing between them. CPGNN [26] propagates soft labels with the help of a compatibility matrix. This method does not change the traditional propagation mechanism, which also makes the representation of nodes from different categories mixed together and indistinguishable. HOG-GCN [27] constructs a homophily degree matrix using the label propagation technique to investigate the extent to which pair of nodes in a graph with strong heterophily belongs to the same class. BM-GCN [28] creates a new network topology by using block similarity matrix that is commonly used in community detection, allowing it to explore block-guided neighbors and conduct adaptive aggregation. However, these methods coarsely use the label information or other supplement information (e.g., block similarity matrix) to guide the neighbor aggregation. While doing so can merge effective information to some extent, it also introduces noise and irrelevant information, which harms the prediction of downstream tasks. Most significantly, as the number of layers increases, the noise and error become increasingly large, rendering the representation of nodes from different classes indistinguishable and causing the model to suffer from severe over-smoothing issue. Furthermore, these methods tend to face the severe performance degradation, when the labels are modified by attackers. In this paper, we propose a label estimation-based message passing to adaptive aggregate neighbors and employ symmetric cross-entropy loss to improve the reliability of estimated information and enhance the robustness to noise labels.

3. Preliminary study

In this section, we start with an introduction of notations and the GCN model. We then introduce the definition of heterophily.

3.1. Notations

We denote matrices with boldface capital letters (e.g. $\boldsymbol{X}$), vectors with boldface lowercase letters (e.g., $\boldsymbol{x}$) and scalars with lowercase alphabets (e.g., x) (Talbe 1). Let $G = (\mathcal{V}, \mathcal{E})$ represent an undirected graph, where $\mathcal{V} = \{v_i\}$ and $\mathcal{E} = \{(v_i, v_j)\}$ denote the node and edge sets, respectively. The node feature matrix is represented by $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, where n represents the number of nodes and

d represents the number of features per node. The graph G is described by the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ that associates each edge (v_i, v_j) with its element A_{ij}; We let $\boldsymbol{D}$ be the degree matrix, let $\tilde{\boldsymbol{A}} := \boldsymbol{A} + \boldsymbol{I}$ and $\tilde{\boldsymbol{D}} := \boldsymbol{D} + \boldsymbol{I}$ be the adjacency and degree matrices of the graph augmented with self-loops. The symmetric normalized adjacency matrix is given by $\hat{\boldsymbol{A}} := \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}}$, which is widely used for spatial neighborhood aggregation in GNN models. The main symbols and notations are summarized on Table 1.

3.2. Graph convolution networks

To learn node representations, GCNs adopt feature transformation and neighbor aggregation techniques to update the representation of nodes by considering the node features and graph connective topology. At the l-th layer, the graph convolutional layer is mathematically expressed as:

$$\boldsymbol{H}^{\ell+1} = \sigma\left(\hat{\boldsymbol{A}} \boldsymbol{H}^{\ell} \boldsymbol{W}^{\ell}\right), \tag{1}$$

where $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\boldsymbol{A}}$. $\sigma(\cdot)$ denotes an non-linear activation function, such as ReLU. $\boldsymbol{H}^0 = \boldsymbol{X}$ is the input feature matrix for the first layer, while $\boldsymbol{H}^{\ell+1}$ is the output embedding matrix from layer ℓ. $\boldsymbol{W}^{\ell}$ is a linear transformation matrix.

The graph convolution layer in GCNs can be split into two steps: $\hat{\boldsymbol{A}} \boldsymbol{H}^{\ell}$ is the aggregation step, and $\boldsymbol{H}^{\ell} \boldsymbol{W}^{\ell}$ is the feature transformation step, from the spatial point of view. The aggregation stage tends to make the local neighborhood's node features similar. Then the feature transformation operation plays the role of feature extractor, easily extracting the commonality between neighboring nodes. Under the assumption of homophily, the above design is excellent. However, in heterogeneous graphs, aggregating without the guidance of label information results in the intermingling of nodes from different classes. This leads to a gradual accumulation of classification errors, especially as the number of layers increases.

3.3. Heterophily

Given a set of node labels, homophily captures the tendency for linked nodes to often belong to the same label, while heterophily is the opposite. The measure of graph homophily level can be defined as follows, which can be used to define graphs with strong homophily/heterophily:

Definition 1. *The homophily ratio* $h = \frac{\left|\{(v_i, v_j) \in \mathcal{E} \wedge y_i = y_j\}\right|}{|\mathcal{E}|}$ *is the proportion of edges in the graph that linked nodes with the same label.*

A high homophily ratio h (e.g., $h \to 1$) indicates the graph with strong homophily, a low homophily ratio h (e.g., $h \to 0$) indicates the graph with strong heterophily. Most previous work considers graphs with strong homophily. However, there are graphs with strong heterophily in the real world, where previous work fails to perform well.

4. The proposed framework

Most GNN models design the message passing mechanisms based on the assumption that linked nodes have same class. As a result, these models can perform well on homophilic graphs. However, heterophilic graphs also exist in the real world, and their linked nodes often belong to different labels. It is challenging for typical GNN models to capture heterophily. It is impossible to distinguish neighbors from different labels based on similar or mixed learned representations. If GNN models attempt to explore larger receptive fields by stacking more layers, this confused representation will continue to be passed on, leading to a more severe over-smoothing issue. H2GNN proposes to separate the ego-embedding from neighbor embedding, which boosts learning from the graph structure under heterophily. It is a simple way to prevent mixing the embeddings from neighbors results in final embeddings indistinguishable. In this work, we have the following research question: can the typical message passing mechanism be modified to aggregate neighbors based on labels to capture the heterophily and easily go deeper?

4.1. Label estimation-based GNN

To answer this question, we propose a novel Label Estimation-based Graph Neural Network named LE-GNN (Fig. 2). Mainly, LE-GNN focuses on aggregating neighbors with the same label by an optimal adjacency matrix, making the class boundaries clear and easy to learn distinguishable node representations. The key to enhancing GCN performance is to make edge weights trainable, allowing nodes in the same class / label to connect more strongly with each other. It is worth noticing how to get an optimal adjacency matrix. Specifically, we first predict the labels using a label estimator, and we use this label prediction matrix to compute the node label similarity matrix. We then get the optimal edge weights based on label information. To learn node representations and perform the final node classification, we input ideal edge weights into the GCN, aggregating neighbors with consistent labels. As introduced in the following, LE-GNN consists of three key components: label estimation, enhancing robustness to noisy labels, label estimation-based message passing. The framework is illustrated in Figure 2.

Label estimation. Nodes close to each other in the feature space will tend to share identical label in the label space [29]. In other words, nodes have same label tend to have consistent distribution. Therefore, original features have some kind of correlation with

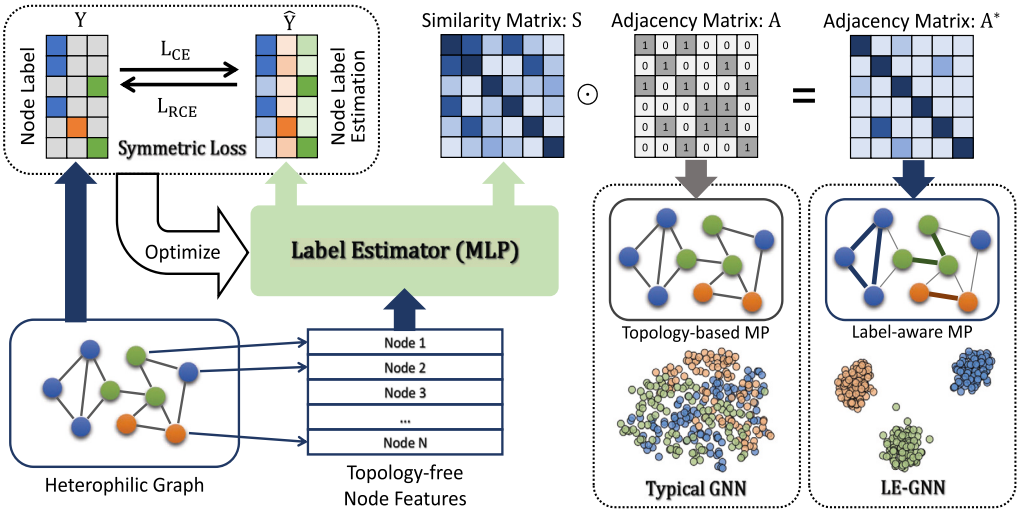

Fig. 2. An illustration of our framework LE-GNN. MLP is employed to estimate unlabeled nodes. With those estimated labels and ground-truth, we obtain a new adjacency matrix $\boldsymbol{A}^*$ with label information. Then, $\boldsymbol{A}^*$ is used for message passing.

the labels, which is the key to the success of the neural network model. Even simple Multi-Layer Perceptron (MLP) can perform better than GCN in some cases. To obtain the preliminary estimated label, we use an effective label estimation method to map the node attributes of feature space into their associated label space. For this paper, MLP is used to estimate the labels of nodes i with i's features, which does not depend on graph structure. Label estimation is expressed as:

$$\hat{\boldsymbol{y}}_i^E = \mathrm{MLP}\left(\boldsymbol{x}_i\right), \tag{2}$$

where $\boldsymbol{x}_i$ is the features of node i. After projection, we get the label estimation results $\hat{\boldsymbol{Y}}^E \in \mathbb{R}^{n\times c}$, where each row of $\hat{\boldsymbol{Y}}^E$ is a probability distribution deriving from the softmax. We train the MLP to minimize the loss function:

$$\mathcal{L}_{\mathrm{CE}} = \arg\min \frac{1}{|\mathcal{V}_{\mathrm{train}}|} \sum_{i\in\mathcal{V}_{\mathrm{train}}} \mathbb{E}\left(\hat{\boldsymbol{y}}_i^E, \boldsymbol{y}_i\right), \tag{3}$$

where $\mathcal{V}_{\mathrm{train}}$ is set of nodes in the training set, $\mathbb{E}(\cdot)$ represents cross entropy loss, $\hat{\boldsymbol{y}}_i^E$ denotes estimated label of node i and $\boldsymbol{y}_i$ denotes the true label. The goal of optimization is to make the two distributions as similar as possible, i.e., to minimize the cross entropy.

Enhancing robustness to noise labels. In label estimation, we use Cross Entropy (CE) to improve the prediction accuracy of sample nodes corresponding to the correct labels during the MLP training. CE by itself is not sufficient for learning of classes, especially under the noisy label scenario. To reduce the effect of noisy labels, the simplest method is to use label smoothing regularization to mitigate the effect of over-fitting due to this noise. However, this approach did not achieve significant improvement [30]. Wang et al. [30] argues that DNNs learned with cross entropy tend to exhibit over-fitting to noisy labels on long-tailed classes, and it suffers greatly in learning some short-tailed classes. Inspired by [30], we employ a Symmetric Cross Entropy (SCE) approach to learn robustly under noisy labels to improve the accuracy of prediction. The Symmetric Cross Entropy (SCE) denoted by $\mathcal{L}_{SCE}$ is defined as

$$\mathcal{L}_{\mathrm{SCE}} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{RCE}} = \mathbb{E}(\boldsymbol{Q}, \boldsymbol{P}) + \mathbb{E}(\boldsymbol{P}, \boldsymbol{Q}), \tag{4}$$

where $\boldsymbol{P}$ denotes the ground-truth distribution of samples, and $\boldsymbol{Q}$ denotes its fitted distribution via the model. The inherent assumption of Symmetric Cross Entropy is that if $\boldsymbol{P}$ includes noisy labels due to mistakes from annotations or attacks, $\mathbb{E}(\boldsymbol{Q}, \boldsymbol{P})$ that observes Q from $\boldsymbol{P}$ may be insignificant; reversing the observation direction, that is, adding the constrain of $\mathbb{E}(\boldsymbol{P}, \boldsymbol{Q})$ may enhance the modeling of entropy on $\boldsymbol{P}$ and $\boldsymbol{Q}$. Specifically, $\mathcal{L}_{\mathrm{RCE}}$ is defined as:

$$\mathcal{L}_{\mathrm{RCE}} = \arg\min \frac{1}{|\mathcal{V}_{\mathrm{train}}|} \sum_{i\in\mathcal{V}_{\mathrm{train}}} \mathbb{E}\left(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i^E\right). \tag{5}$$

Recalling $\mathcal{L}_{\mathrm{CE}}$ of Eq. (3), with Eq. (5), Eq. (4) is then written as

$$\mathcal{L}_{\mathrm{SCE}} = \lambda_1 \mathcal{L}_{\mathrm{CE}} + \lambda_2 \mathcal{L}_{\mathrm{RCE}}, \tag{6}$$

where λ_1 and λ_2 are hyperparameters.

Label estimation based message passing. The typical message passing mechanism without the guide of label information, which mixes the neighbors with different labels, delivers low-quality neighbor information on heterophilic graph. In this part, we unify the label, feature, and graph structure of the GNN model by replacing typical aggregation with label estimation-based message

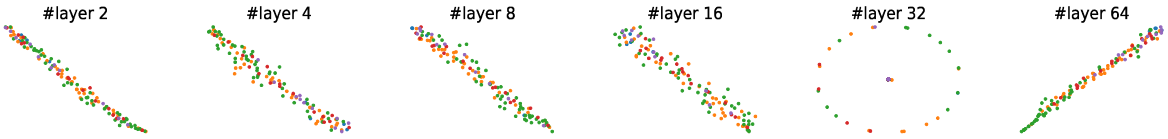

Fig. 3. Visualization of GCN with varying layers on Wisconsin. Node classes are drawn in different colors.

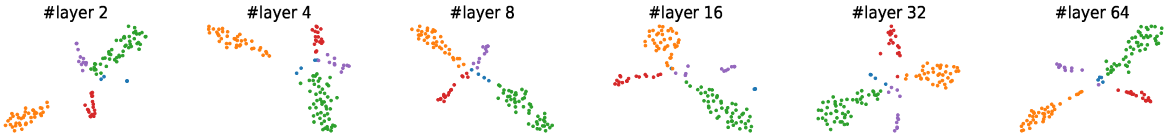

Fig. 4. Visualization of LE-GNN with varying layers on Wisconsin. Node classes are drawn in different colors.

passing. After performing label estimation, we can obtain the predicted labels of the unlabeled nodes. Combining them with the provided ground-truth $\boldsymbol{Y}_{train}$, we have labels $\tilde{\boldsymbol{Y}}^{E} = \hat{\boldsymbol{Y}}^{E} \cup \boldsymbol{Y}_{train}$. Given these labels, we will obtain a label similarity matrix $\boldsymbol{S} = \tilde{\boldsymbol{Y}}^{E}\tilde{\boldsymbol{Y}}^{E^{\top}}$, $\tilde{\boldsymbol{Y}}^{E} \in \mathbb{R}^{n\times c}$ is a set of soft labels, representing the probability that the node belongs to the corresponding label. The larger $\boldsymbol{S}_{i,j}$ is, the more consistent the labels of node i and node j are. We perform a Hadamard product of the label similarity matrix $\boldsymbol{S}$ and adjacency matrix $\boldsymbol{A}$. In this way, the elements of $\boldsymbol{A}$ will be changed according to the $\boldsymbol{S}$. The larger $\boldsymbol{S}_{i,j}$ is, the larger $\boldsymbol{A}_{i,j}$ is. This means that the more consistent the labels of the connected nodes are, the larger the corresponding edge weights will be. In other words, this optimal matrix will retain more information about the node's neighbors with consistent labels. It can be expressed as:

$$\boldsymbol{A}^{*} = \boldsymbol{S} \odot \boldsymbol{A}, \tag{7}$$

where $\boldsymbol{A}$ is original adjacency matrix, $\boldsymbol{A}$ is always optimized since $\boldsymbol{S}$ is learnable. Instead of adding or removing edges to change the basic structure of the graph, we change the weights of the edges by learning the optimal similarity matrix. Finally, unlike traditional message passing, we conduct label estimation based message passing by using $\boldsymbol{A}^{*}$:

$$\boldsymbol{H}^{(\ell+1)} = (1-\alpha)\boldsymbol{A}^{*}\boldsymbol{H}^{(\ell)} + \alpha\boldsymbol{H}^{0}, \tag{8}$$

where α is a factor hyperparameter to control the trade-off between the current convolutional layer and the input layer. Instead of separating the ego-embedding from neighbor embedding, we set up an initial connection. This simple operation preserves the original representation $\boldsymbol{H}^{0}$, which prevents the loss of initial node attributes due to the iterative aggregation of neighbor information. Most important is performing message passing based on $\boldsymbol{A}^{*}$, which makes more information with the same labels flow to the central nodes.

Overall objective function Besides the above-mentioned loss function $\mathcal{L}_{\text{SCE}}$ for label estimation, we introduce the loss function of label estimation based message passing (LEMP). We denote the output of the last LEMP layer as $\hat{\mathbf{H}}_{out}$, then the classification loss is shown as,

$$\mathcal{L}_{\text{LEMP}} = \arg\min \frac{1}{|\mathcal{V}_{\text{train}}|} \sum_{i\in\mathcal{V}_{\text{train}}} \mathbb{E}\left(\text{softmax}\left(\hat{\mathbf{H}}_{out}\right), \boldsymbol{y}_i\right). \tag{9}$$

Therefore, the overall objective function can be defined as follows:

$$\begin{aligned} \min \mathcal{L} &= \mathcal{L}_{\text{LEMP}} + \mathcal{L}_{\text{SCE}} \\ &= \mathcal{L}_{\text{LEMP}} + \lambda_1\mathcal{L}_{\text{CE}} + \lambda_2\mathcal{L}_{\text{RCE}}, \end{aligned} \tag{10}$$

where λ_1 and λ_2 are hyper-parameters to trade off the ability of prediction and noise resistance.

4.2. Towards preventing over-smoothing

Despite the great success in modeling graph data of GNNs, a key limitation of GNNs is over-smoothing. This problem occurs when GNNs go deeper to explore the larger receptive field. Due to the recursive neighbor aggregation by adding more GNN layers, the node representations become indistinguishable vectors, which significantly decreases the performance of GNNs. Recalling Eq. (1), $\hat{\boldsymbol{\lambda}}\boldsymbol{H}^{\ell}$ is the aggregation process of ℓ-th layer in typical GNNs, which can be written as:

$$\boldsymbol{h}_{v}^{(\ell)} = \text{AGGREGATE}^{(\ell)}\left(\left\{\boldsymbol{h}_{u}^{(\ell-1)} : u \in \mathcal{N}(v)\right\}\right), \tag{11}$$

where $\text{AGGREGATE}^{(\ell)}(\cdot)$ represents the aggregation operation in the ℓ-th layer, $\mathcal{N}(v)$ denotes the linked neighbors of node v. This traditional aggregation makes model challenging to separate the learned representations for heterophilic graphs due to ignoring the label information. As shown in Fig. 3, the node representations become indistinguishable when several GCN layers are deployed, which limits the model to explore a larger receptive field to take advantage of more useful neighbor information. To prevent over-smoothing issue, we propose a novel label estimation-based message passing to aggregate neighbor nodes. The goal is to obtain an optimal adjacency matrix with label information. The novel aggregation process of ℓ-th layer can be represented as:

Table 2
Summary of datasets used in node classification. h denotes the homophily ratio.

Dataset	Classes	Nodes	Edges	Features	h
Cora	7	2,708	5,429	1,433	0.81
Citeseer	6	3,327	4,732	3,703	0.74
Pubmed	3	19,717	44,338	500	0.8
Cornell	5	183	295	1,703	0.3
Texas	5	183	309	1,703	0.11
Wisconsin	5	251	499	1,703	0.21
Squirrel	5	5,201	19,8353	2,089	0.21
Chameleon	5	2,277	3,1371	2,325	0.21

Table 3
Accuracy (%) comparisons with SOTA on homophilic graphs. The highest performances are in bold.

Method	Cora	Citeseer	Pubmed
GCN	81.5	71.1	79.0
GAT	83.1	70.8	78.5
APPNP	83.3	71.8	80.1
JKNet	81.9	69.8	78.1
JKNet(Drop)	83.3	72.6	79.2
Incep(Drop)	83.5	72.7	79.5
SGC	81.7	71.7	78.9
DAGNN	84.4±0.5	73.3±0.6	80.5±0.5
SimP-GCN	82.8±0.4	72.6±0.5	**81.1±0.3**
LE-GNN	**85.6± 0.5**	**73.9± 0.6**	79.9±0.2

$$\hat{\boldsymbol{h}}_v^{(\ell)} = \text{AGGREGATE}^{(\ell)}\left(\left\{\boldsymbol{h}_u^{(\ell-1)} : u \in \hat{\mathcal{N}}(v)\right\}\right), \tag{12}$$

where $\hat{\mathcal{N}}(v) = \left\{u : (v,u) \in \mathcal{E} \wedge \hat{\boldsymbol{y}}_u^E = \hat{\boldsymbol{y}}_v^E\right\}$ is node v's neighbors with same label as the node v. Unlike typical aggregation makes node representations from different clusters become mixed up, our label estimation-based message passing has the ability to adaptively control the contribution of each node by considering the label information. In each connected subgraph, edges consisting of connected nodes with same label will have larger weights, which prevents nodes from mixing across clusters. Cong et al. [31] holds that adding to the number of layers causes over-smoothing, which makes all node representations converge to the same point, losing the dissimilarity (i.e., distance) between nodes. The following Lemma demonstrates that the aggregation step minimizes the overall distance in the embedding space between the graph's connected nodes:

Lemma 1. *Let* $D(\boldsymbol{x}) = \frac{1}{2}\sum_{v_i,v_j}\tilde{a}_{ij}\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2$ *be a distance metric over node embeddings* $\mathbf{x}$. *Then we have*

$$D\left(\boldsymbol{h}^{(\ell)}\right) \le D\left(\boldsymbol{x}^{(\ell)}\right), \tag{13}$$

where $\boldsymbol{h}^{(\ell)}$ *denotes the hidden representation in the* ℓ*-th layer.*

The Lemma 1 has been proofed by [32]. It indicates that the distance between nodes will become smaller after one layer of graph convolution because the aggregation operation will squeeze the representation space of the nodes, forcing the neighbor nodes to move closer to the central node. The ideal situation is that neighbors with the same label are close to the central node, while neighbors with different labels are far away from the central node, and in this way, the inter-class boundaries become clearer. Our approach achieves the goal of preserving more information of nodes with the same label precisely by controlling edge weights, mitigating the negative impact of noisy neighbors, which reveals that our approach has the ability to capture heterophily and can go deeper. To verify the effectiveness of LE-GNN, we utilize the t-SNE to provide interpretable visualizations on Wisconsin that is a heterophilic graph. The t-SNE is able to capture the clustering results in high-dimensional data, which is consistent with the classification of nodes. As shown in Fig. 4, our model with a large receptive field, such as 64-hop, still generates distinguishable node representations, suggesting that our LE-GNN is able to prevent over-smoothing.

5. Experiments

In this section, we conduct extensive experiments to evaluate our model and aim at answering the following questions:

- **Q1**: How does LE-GNN perform on homophilic graphs?
- **Q2**: How effective is LE-GNN compared with competitive baselines in the node classification task on the graphs with heterophily?
- **Q3**: Whether LE-GNN could tame the over-smoothing representation on heterophilic graphs?

Table 4
The hyper-parameters for Table 6.

Dataset	Method	Hyper-parameters
Cora	GCNII	layers: 64, α_ℓ: 0.2, lr: 0.01, hidden: 64, λ: 0.5, dropout: 0.5, L_2: 1e-4
	HOG-GCN	layers: 3, lr: 0.001, hidden: 256, dropout: 0.5, L_2: 5e-4
	BM-GCN	layers: 3, lr: 0.001, hidden: 64, dropout: 0.5, enhance: 3.0, self_loop: 1.5, L_2: 5e-4
	LE-GNN	layers: 16, α: 0.2, lr: 0.01, hidden: 128, λ_1: 1e-3 , λ_2: 1e-2, dropout: 0.5, L_2: 5e-4
Cite.	GCNII	layers: 64, α_ℓ: 0.5, lr: 0.01, hidden: 64, λ: 0.5, dropout: 0.5, L_2: 5e-6
	HOG-GCN	layers: 3, lr: 0.001, hidden: 256, dropout: 0.5, L_2: 5e-4
	BM-GCN	layers: 3, lr: 0.001, hidden: 64, dropout: 0.5, enhance: 4.0, self_loop: 2.0, L_2: 5e-4
	LE-GNN	layers: 8, α: 0.3, lr: 0.01, hidden: 64, λ_1: 1e-3 , λ_2: 1e-2, dropout: 0.5, L_2: 5e-4
Pubm.	GCNII	layers: 64, α_ℓ: 0.1, lr: 0.01, hidden: 64, λ: 0.5, dropout: 0.5, L_2: 5e-6
	HOG-GCN	layers: 3, lr: 0.001, hidden: 256, dropout: 0.5, L_2: 5e-4
	BM-GCN	layers: 3, lr: 0.001, hidden: 64, dropout: 0.5, enhance: 2.0, self_loop: 3.0, L_2: 5e-4
	LE-GNN	layers: 2, α: 0.2, lr: 0.005, hidden: 256, λ_1: 1e-3, λ_2: 1e-2, dropout: 0.5, L_2: 5e-5
Cham.	GCNII	layers: 8, α_ℓ: 0.2, lr: 0.01, hidden: 64, λ: 1.5, dropout: 0.5, L_2: 5e-4
	HOG-GCN	layers: 3, lr: 0.001, hidden: 256, dropout: 0.5, L_2: 5e-4
	BM-GCN	layers: 3, lr: 0.001, hidden: 64, dropout: 0.5, enhance: 0.8, self_loop: 0.0, L_2: 5e-4
	LE-GNN	layers: 1, α: 0, lr: 0.01, hidden: 64, λ_1: 1e-3 , λ_2: 1e-2, dropout: 0.2, L_2: 5e-4
Corn.	GCNII	layers: 16, α_ℓ: 0.5, lr: 0.01, hidden: 64, λ: 1, dropout: 0.5, L_2: 0.001
	HOG-GCN	layers: 3, lr: 0.005, hidden: 256, dropout: 0.5, L_2: 0.02
	BM-GCN	layers: 2, lr: 0.001, hidden: 64, dropout: 0.5, enhance: 1.0, self_loop: 0.0, L_2: 5e-4
	LE-GNN	layers: 8, α: 0.9, lr: 0.01, hidden: 128, λ_1: 1e-3 , λ_2: 1e-2, dropout: 0.5, L_2: 1e-5
Texa.	GCNII	layers: 32, α_ℓ: 0.5, lr: 0.01, hidden: 64, λ: 1.5, dropout: 0.5, L_2: 0.0001
	HOG-GCN	layers: 3, lr: 0.005, hidden: 256, dropout: 0.5, L_2: 0.02
	BM-GCN	layers: 2, lr: 0.001, hidden: 64, dropout: 0.5, enhance: 1.0, self_loop: 0.0, L_2: 5e-4
	LE-GNN	layers: 2, α: 0.9, lr: 0.01, hidden: 64, λ_1: 1e-3 , λ_2: 1e-2, dropout: 0.5, L_2: 1e-5
Wisc.	GCNII	layers: 16, α_ℓ: 0.5, lr: 0.01, hidden: 64, λ: 1, dropout: 0.5, L_2: 0.0005
	HOG-GCN	layers: 3, lr: 0.005, hidden: 256, dropout: 0.5, L_2: 0.02
	BM-GCN	layers: 2, lr: 0.001, hidden: 64, dropout: 0.5, enhance: 1.0, self_loop: 0.0, L_2: 5e-4
	LE-GNN	layers: 2, α: 0.9, lr: 0.01, hidden: 128, λ_1: 1e-3 , λ_2: 1e-2, dropout: 0.5, L_2: 1e-5
Squi.	GCNII	layers: 16, α_ℓ: 0.5, lr: 0.01, hidden: 64, λ: 1, dropout: 0.5, L_2: 0.0005
	HOG-GCN	layers: 3, lr: 0.005, hidden: 256, dropout: 0.5, L_2: 0.02
	BM-GCN	layers: 3, lr: 0.001, hidden: 64, dropout: 0.5, enhance: 2.0, self_loop: 0.0, L_2: 5e-4
	LE-GNN	layers: 2, α: 0.2, lr: 0.01, hidden: 64, λ_1: 1e-3 , λ_2: 1e-2, dropout: 0.5, L_2: 1e-4

Specifically, we first verify the performance of the model on homophilic and heterophilic graphs, then conduct robustness analysis for LE-GNN, and finally consider the ablation experiments of the model. Code is available at https://github.com/KaiGuo20/LE-GNN.

5.1. Datasets and experiment settings

Benchmark datasets. We now evaluate the performance of our model and existing GNNs on a variety of real-world datasets [33–36]. According to the homophily ratio $\boldsymbol{h}$, the graphs can be classified into homophilic and heterophilic graphs. We use three homophilic graphs: Cora, Citeseer and Pubmed, and five heterophilic graphs: Cornell, Texas, Wisconsin, Squirrel and Chameleon to evaluate the performance. The statistics of datasets are presented in Table 2, including the information about homophily ratio.

Experimental settings. We implement LE-GNN and reproduce the baselines using Pytorch [37] and Pytorch Geometric [38]. To perform a rigorous and fair comparison of the different models on each dataset, we use the same dataset splits and training procedures. Specifically, we follow the dataset partitioning proposed by [39] for experiment on homophilic graphs, and follow the dataset partitioning proposed by Geom-GCN [20] for experiment on heterophilic graphs. We calculate the average test accuracy of 10 runs for the node classification. For experiment of deep model and adversarial robustness, we report the average performance of 5 runs. We adopt the Adam optimizer [40] for model training. The seeds of random numbers are set to the same value for the purpose of reproducibility. We perform random hyper-parameter searches and report the case that achieves the best accuracy on the validation set of each benchmark. For our proposed LE-GNN, we tune the following hyper-parameters: $\alpha \in \{0, 0.1, 0.2, ..., 1\}$, $\lambda_1 \in \{5e-4, 1e-3, 1e-2\}$, $\lambda_2 \in \{5e-4, 1e-3, 1e-2\}$. The hyper-parameters have been reported on Table 4.

5.2. Baselines

We compare the proposed model with the following representative state-of-the-art methods:

GCN [14]. GCN is the first notable GNN graph embedding research. To learn graph embedding, GCN uses aggregation and feature transformation functions to encode both graph connection structure and node attributes.

Table 5
Testing mean accuracy (%) results with various layers on homophilic graphs.

Dataset	Method	Layers					
		2	4	8	16	32	64
Cora	GCN	81.52±0.71	78.12±1.67	63.11±7.47	41.05±17.92	17.96±7.36	21.14±9.35
	JKNet	81.87±0.75	80.59±1.13	79.50±1.29	80.21±1.36	81.12±1.41	71.54±1.27
	LE-GNN	81.90±0.49	83.77±0.29	83.85±0.55	85.60±0.51	85.10±0.37	84.50±0.38
Citeseer	GCN	71.15±1.49	62.15±1.68	53.76±5.45	30.05±6.93	18.47±2.58	17.94±2.01
	JKNet	70.38±1.22	70.47±0.45	70.30±1.09	69.81±1.02	68.25±0.98	63.47±1.08
	LE-GNN	71.79±0.50	71.99±0.95	73.25±0.54	73.92±0.61	73.27±0.81	73.30±1.04

GAT [17]. GCN aggregates feature information of neighbors using laplacian matrix, which assigns the same weight to different neighbors. In order to address this issue, the Graph Attention Network (GAT) uses an attention mechanism and learns different scores to various nodes within the neighborhood.
APPNP [16]. APPNP combines GCN and PageRank to obtain an improved personalized PageRank-based propagation scheme that incorporates higher-order neighborhood information while preserving local information.
DropEdge [24]. DropEdge eliminates edges from the original graph at random during model training to mitigate problems like over-smoothing and over-fitting induced by stacking more GCN layers.
JKNet [23]. JKNet learns better representations by leveraging diverse neighbor ranges for each node.
DAGNN [41]. To tackle over-smoothing issue, DAGNN decouples two operations: representation transformation and propagation to learn graph node representations from larger receptive fields.
GCNII [21]. In order to create a deep GNN model for better performance, GCNII incorporates residual connectivity and identity mapping on the basis of GCN.
H2GNN [15]. H2GCN adopts three key designs for node classification on heterophilic graphs.
Geom-GCN [20]. Geom-GCN explores to capture long-range dependencies in disassortative graphs. It constructs structural neighborhoods for aggregation using the geometric relationships defined in the latent space.
TDGNN [25]. The Tree Decomposed Graph Neural Network (TDGNN) is able to adaptably incorporate data from large receptive fields and aggregate this information with the help of multi-hop dependency.
SimP-GCN [42]. SimP-GCN adaptively integrates the graph structure and node features by using a feature similarity preserving aggregation. In addition, it captures pairwise node similarity via self-supervised learning.
CPGNN [26]. CPGNN propagates soft labels with the help of a compatibility matrix.
HOG-GCN [27]. HOG-GCN constructs a homophily degree matrix using the label propagation technique to investigate the extent to which pair of nodes in a graph with strong heterophily belongs to the same label.
BM-GCN [28]. BM-GCN creates a new network topology by using block similarity matrix that is commonly used in community detection.

To investigate the applicability of LE-GNN, we expand our empirical evaluation to two types of graphs: homophilic graphs and heterophilic graphs.

5.3. Performance comparison on homophilic graphs

To response the **Q1**, we conduct experiments on homophilic graphs. There are three well-known homophilic benchmark datasets: Cora [33], Citeseer [33] and Pubmed [33] for semi-supervised node classification.

Settings. To verify the effectiveness of LE-GNN on homophilic graphs, we conduct two experiments, namely the comparison with different layers and comparison with SOTA. We follow the standard experimental setting proposed by [39]. We consider the following baselines: GCN [14], GAT [17], APPNP [16], JKNet [23], DropEdge [43], SGC [18], DAGNN [41], SimP-GCN [42]. To ensure a fair comparison, we use the same training/validation/test splits as in DAGNN for each model. We compute the average test accuracy of 10 runs.

The results of LE-GNN compared with SOTA methods are reported in Table 3. We observe that our proposed LE-GNN outperforms the SOTA methods on 2 out of 3 datasets, which demonstrates the effectiveness of our model on homophilic graph. Quantitatively, LE-GNN obtains the improvement of 1.4% and 0.82% on Cora and Citeseer, respectively. Table 5 summarizes the results of comparison with different layers. We find that the performance of our model LE-GNN keeps improving or stabilizing as the number of layers increases on homophilic graphs like Cora and Citeseer.

5.4. Performance comparison on heterophilic graphs

It is challenging for graph neural networks to perform node classification on heterophilic graphs. In the following experiments, we evaluate the performance of LE-GNN on graphs with heterophily.

Settings. In order to provide a rigorous and fair comparison between the different models on each dataset, the dataset split and training procedures are set to the same. Following the common setting proposed by Geom-GCN [20], we randomly split the nodes of each class into 60%, 20% and 20% for training, validation and testing, respectively, for each dataset.

Table 6
Accuracy (%) comparisons with SOTA over 8 datasets. The highest performances are in bold.

Method	Cora	Cite.	Pumb.	Corn.	Texa.	Wisc.	Squi.	Cham.
GCN	85.77	73.68	87.91	52.70	58.38	39.02	36.89	59.82
GAT	86.37	74.32	87.62	54.32	58.38	49.41	30.62	54.69
APPNP	87.87	76.53	89.40	73.51	65.41	69.02	30.67	43.85
JKNet	85.25	75.85	88.94	57.30	56.49	48.82	40.45	60.07
Incep(Drop)	86.86	76.83	89.18	61.62	57.84	50.20	41.78	61.71
DAGNN	87.26±1.42	76.47±1.54	87.49±0.63	80.97±6.33	81.32±4.19	85.38±3.95	36.60±1.25	50.75±3.46
GCNII*	88.27 ± 1.31	77.06±1.67	**90.26±0.41**	76.70±5.40	77.57±5.84	80.39±4.94	38.49±1.58	62.48±1.25
Geom-GCN*	85.35±1.57	**78.02±1.15**	90.05 ± 0.47	60.54±3.67	66.76±2.72	64.51±3.66	38.15±0.92	60.90±1.59
H2GCN*	87.81±1.35	77.11±1.64	89.49±0.34	82.16±4.80	84.86 ± 6.77	86.67 ± 4.69	37.90±2.02	59.39±1.98
TDGNN*	88.26±1.32	76.64±1.54	89.22±0.41	82.92±6.61	83.00±4.50	85.57±3.78	-	-
SimP-GCN	85.56±1.87	74.64±1.10	89.90±0.22	84.05 ± 6.40	81.62±4.88	85.49±3.24	41.59±2.67	53.79±0.82
CPGNN-MLP	85.23±1.71	74.29±2.41	86.83±0.78	-	77.09±4.22	84.53±6.48	28.65±1.50	52.63±1.79
CPGNN-Cheby	86.82±1.11	75.42±1.85	89.08±0.67	-	77.03±5.83	81.76±6.74	30.95±1.24	54.05±4.67
HOG-GCN	86.27±1.77	76.30±1.94	88.35±0.50	77.56±6.11	81.62±4.88	81.37±3.36	37.85±1.55	53.99±1.35
BM-GCN	87.72±0.87	76.54±1.77	89.01±0.57	74.89±5.10	80.81±7.14	76.47±5.91	**53.12±1.67**	**69.12±1.74**
LE-GNN	**88.39±1.58**	77.47 ± 1.36	89.63±0.41	**84.59±5.72**	**86.22±5.47**	**86.86±3.55**	45.05 ± 1.34	65.42 ± 1.97

Table 7
Testing mean accuracy (%) results with various layers on heterophilic graphs.

Dataset	Method	Layers					
		2	4	8	16	32	64
Chameleon	GCN	53.98±2.29	53.26±1.68	38.61±3.75	35.39±3.23	35.20±2.29	35.50±3.15
	Geom-GCN*	61.25±2.21	20.55±1.42	19.58±1.73	19.58±1.73	19.58±1.73	19.58±1.73
	GCNII*	61.69±2.11	61.92±2.54	62.10±1.29	62.48±1.45	57.52±1.62	55.36±1.52
	H2GNN*	59.86±1.56	60.77±1.74	OOM	OOM	OOM	OOM
	BM-GCN	59.86±1.56	64.38±2.48	22.76±2.56	19.58±1.82	20.21±2.98	19.58±1.82
	HOG-GCN	53.99±1.35	42.67±2.39	35.57±1.68	21.05±2.75	22.50±1.51	OOM
	LE-GNN	63.05±2.51	54.41±2.55	53.07±2.73	53.64±2.67	52.59±3.05	53.20±2.75
Wisconsin	GCN	63.59±5.40	54.92±5.42	51.33±6.84	47.84±9.86	47.84±9.86	47.84±9.86
	Geom-GCN*	64.51±3.66	17.86±1.26	15.22±2.67	15.22±2.67	15.22±2.67	15.22±2.67
	GCNII*	78.62±0.18	77.85±1.28	75.27±1.29	73.73±5.29	70.20±5.61	70.98±7.25
	H2GNN*	87.65±4.98	84.57±2.98	80.39±3.57	OOM	OOM	3.92±1.02
	BM-GCN	76.47±5.91	65.09±6.17	47.45±8.92	3.13±1.07	3.13±1.07	3.13±1.07
	HOG-GCN	81.37±3.36	63.53±2.63	48.63±9.13	47.84±9.86	47.84±9.86	47.84±9.86
	LE-GNN	86.86±3.55	84.71 ±1.18	85.10±3.60	86.64±3.67	85.49±3.24	85.15±2.14
Cornell	GCN	60.62±5.40	58.68±5.42	57.84±6.84	57.84±6.84	57.84±6.84	57.84±6.84
	Geom-GCN*	61.02±3.57	21.98±2.61	12.97±2.91	12.97±2.91	12.97±2.91	12.97±2.91
	GCNII*	74.05±4.58	75.14±1.28	75.14±5.29	77.59±4.67	75.36±5.10	73.15±4.36
	H2GNN*	81.67±5.20	81.90±4.52	80.18±3.90	OOM	OOM	OOM
	BM-GCN	74.89±5.10	61.08±8.46	57.30±3.52	11.89±2.41	11.89±2.41	11.89±2.41
	HOG-GCN	77.56±6.11	64.32±8.63	57.83±3.08	57.83±3.08	57.83±3.08	57.83±3.08
	LE-GNN	84.05±5.90	83.97±4.98	84.59±5.72	84.41±6.26	84.59±8.20	84.32±5.26

Overall results In order to verify the overall performance of our proposed model LE-GNN on graphs with homophily and heterophily, we compare LE-GNN with current SOTA methods. For GCN, GAT, and H2GNN*, we reuse the results already provided in [15], as well as the best metrics reported in [25] for Geom-GCN*, GCNII*, and TDGNN*. The results are reported in Table 6, which demonstrates the effectiveness of LE-GNN. We make the following conclusions:

i) The experimental results show that our proposed LE-GNN performs better than baselines on most of the datasets. In particular, for the Texas dataset, our model achieves a significant improvement of 1.6% compared to the second-best model.

ii) Given the results from the Table 6, we find that GCN fails to capture heterophily because it aggregates neighbors without considering the labels of nodes. Similarly, GAT lacks the ability to perform well on the heterophilic graph, although the attention mechanism is used to aggregate the important neighbors' information.

iii) Our LE-GNN adopts a novel message passing mechanism based on label estimation that performs well both on homophilic and heterophilic graphs. The reason is that the scheme aggregates high-quality neighbor information and reduces the negative impact of inconsistently labeled neighbors. The above is the answer to **Q2**.

Validation of taming over-smoothing issue. To answer the **Q3**, we compare our model with baselines under different layers. The results for models with varying number of layers are summarized in Table 7. The following are our observations:

i) Geom-GCN, HOG-GCN and BM-GCN suffer from a severe over-smoothing issue, as evidenced by their dramatic performance degradation after stacking multiple layers. For example, compared to a 2-layer GCN, the prediction accuracy of a 64-layer GCN on

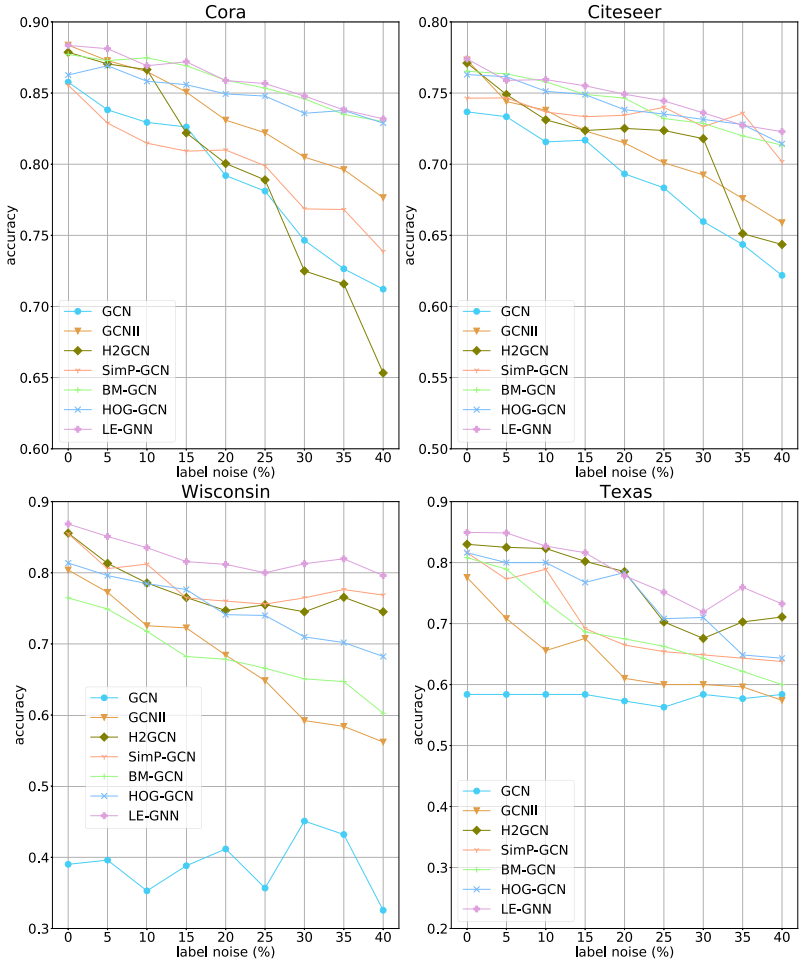

Fig. 5. The model robustness to label noise of different models.

Chameleon is reduced by 34.8%, which demonstrates traditional message passing induces severe over-smoothing issue on heterophilic graphs.

ii)In contrast to representative baseline models, our proposed model generally achieves better performance both on homophilic and heterophilic graphs under various layers. For example, our LE-GNN obtains improvement of 3.0% over H2GNN* with two layers on Cornell dataset, while achieving remarkable gains of 15.3% over GCNII* in the case of 64 layers. For Wisconsin dataset, our proposed LE-GNN delivers remarkable gains of 77.9% over GCN in the case of 64 layers, which is consistent with our previous observation of t-SNE visualization. It demonstrates the effectiveness of our model for preventing the over-smoothing problem. Because our model propagates high-quality neighbor information, avoiding the generation of indistinguishable node representations.

5.5. Robustness analysis for LE-GNN

It is challenging to collect large amounts of data with clean labels for supervised neural network training. This is because the process of data labeling by humans in practice is difficult and expensive, especially in domains that require expert annotation. In addition, some studies have concluded that GNNs are vulnerable to attacks. Once the labels are modified by attackers, the performance of GNNs is greatly degraded [42,44,45]. Moreover, our proposed model relies on the label information for message passing. Therefore, in this subsection, we are interested in investigating its potential benefits in terms of adversarial label robustness. Specifically, we replace the correct label with another label to simulate a label attack. The number of replaced labels divided by the total number of labels is called the label noise ratio. We compare our proposed LE-GNN with some SOTA methods under different label ratios, from 0 to 40% with steps of 5%. We conduct experiments on two heterophilic graphs: Wisconsin, Texas, and two homophilic graphs: Cora, Citeseer.

As shown in Fig. 5, we can observe that LE-GNN consistently improves GCN's performance under varied perturbation rates of adversarial attack. Even there is a high ratio of label noise, LE-GNN enhances GCN by a greater margin. For example, LE-GNN delivers an improvement of 69.6% over GCN under 30% label noise ratio on Wisconsin dataset, which demonstrates that our proposed model

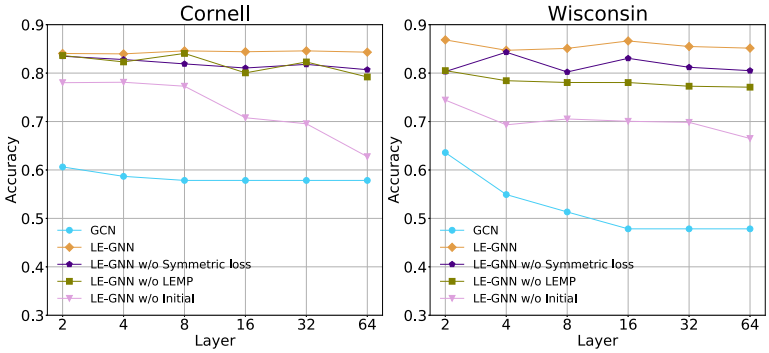

Fig. 6. Ablation study: Depth vs. Accuracy for GCN, our model and three ablated models on Cornell and Wisconsin.

LE-GNN is robust to label noise. The reason is that our model generates soft pseudo labels based on the feature space and improves noise robustness via symmetric cross-entropy, so even if the dataset is attacked and the labels are changed, our model still performs well on node classification tasks.

5.6. Ablation study

LE-GNN consists of three components: label estimation, symmetric cross entropy loss and label estimation based message passing. In our framework, label estimation and label estimation based message passing are required to co-exist because the former is a pre-preparation of latter. Besides, in label estimation based message passing, there is a hyperparameter α about initial connection. Therefore, to evaluate the contributions of our techniques, we construct the following variants of our model and define them as follows: 1) LE-GNN w/o SCE denotes the omission of the Symmetric Cross Entropy module from LE-GNN; 2) LE-GNN w/o LEMP indicates that the label estimation-based message passing in LE-GNN is ignored; 3) LE-GNN w/o Initial indicates to remove the initial connection from LE-GNN. To shed light on the contributions of the three perspectives, we report the node classification results of LE-GNN and their variants on Cornell and Wisconsin, as shown in Fig. 6. We present the following observations.

i) Compared with GCN, our proposed LE-GNN and it's variants deliver significant improvements on heterophilic graphs, which demonstrates the effectiveness of our model. This shows the effectiveness of our proposed LEMP to capture heterophily. This scheme propagates high-quality information by aggregating neighbors with the same label, which facilitates the learning of distinguishable node representations on the heterophilic graphs.

ii) We observe that the GCN model has a sharp performance decrease after four layers. However, our model and its variants perform significantly higher performance. In particular, our model consistently performs better as the number of layers increases, demonstrating our model and its variants have the ability to alleviate the over-smoothing issue.

6. Conclusion

Traditional message passing mechanism is proposed under the assumption that networks with strong homophily. This method is unsuitable for heterophilic graphs, which mixes the neighbors of different labels, delivers low-quality neighbor information, making the learned node representation more indistinguishable. In this paper, we proposed Label Estimation-based GNN (LE-GNN), a deep model that tames over-smoothing issue by introducing extra-label information. LE-GNN includes a message-passing scheme based on label estimation, which aggregates high-quality neighbor information and makes inter-class boundaries clearer. The comprehensive experiments demonstrated that our proposed model LE-GNN could effectively capture heterophily, and further achieved significant improvement in the node classification tasks. Even when the graph suffering severe label attacks, our model could still perform better than SOTA methods, which verified its robustness.

7. Acknowledgments

This work was supported in part by National Natural Science Foundation of China (No.62206108, No.61976102, No.U19A2065) and Lixin Outstanding Young Teacher Training Program of Jilin University.

CRediT authorship contribution statement

Kai Guo: Conceptualization, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Xiaofeng Cao:** Supervision, Writing – review & editing. **Zhining Liu:** Writing – review & editing. **Yi Chang:** Data curation, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, J. Tang, DeepInf: social influence prediction with deep learning, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2110–2119.
[2] C. Li, D. Goldwasser, Encoding social information with graph convolutional networks forpolitical perspective detection in news media, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 2594–2604.
[3] A. Fout, J. Byrd, B. Shariat, A. Ben-Hur, Protein interface prediction using graph convolutional networks, in: NIPS, 2017, pp. 6530–6539.
[4] J. Shang, C. Xiao, T. Ma, H. Li, J. Sun, GAMENet: graph augmented memory networks for recommending medication combination, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 1126–1133.
[5] W. Jin, X. Liu, Y. Ma, C. Aggarwal, J. Tang, Towards feature overcorrelation in deeper graph neural networks, https://openreview.net/forum?id=Mi9xQBeZxY5, 2022.
[6] K. Guo, K. Zhou, X. Hu, Y. Li, Y. Chang, X. Wang, Orthogonal graph neural networks, arXiv preprint, arXiv:2109.11338, 2021.
[7] K. Zhou, X. Huang, D. Zha, R. Chen, L. Li, S.-H. Choi, X. Hu, Dirichlet energy constrained learning for deep graph neural networks, Adv. Neural Inf. Process. Syst. 34 (2021).
[8] H. Gao, Z. Wang, S. Ji, Large-scale learnable graph convolutional networks, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1416–1424.
[9] H. Gao, S. Ji, Graph U-Nets, in: International Conference on Machine Learning, PMLR, 2019, pp. 2083–2092.
[10] M. Zhang, Z. Cui, M. Neumann, Y. Chen, An end-to-end deep learning architecture for graph classification, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[11] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, PMLR, 2017, pp. 1263–1272.
[12] J.M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N.M. Donghia, C.R. MacNair, S. French, L.A. Carfrae, Z. Bloom-Ackermann, et al., A deep learning approach to antibiotic discovery, Cell 180 (2020) 688–702.
[13] Y. Wang, A. Abuduweili, Q. Yao, D. Dou, Property-aware relation networks for few-shot molecular property prediction, Adv. Neural Inf. Process. Syst. 34 (2021).
[14] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of ICLR, 2017.
[15] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, D. Koutra, Beyond homophily in graph neural networks: current limitations and effective designs, in: NeurIPS, 2020.
[16] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: graph neural networks meet personalized pagerank, in: 7th International Conference on Learning Representations, OpenReview.net, 2019.
[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[18] F. Wu, A.H.S. Jr., T. Zhang, C. Fifty, T. Yu, K.Q. Weinberger, Simplifying graph convolutional networks, in: Proceedings of ICML, 2019.
[19] S. Suresh, V. Budde, J. Neville, P. Li, J. Ma, Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1541–1551.
[20] H. Pei, B. Wei, K.C. Chang, Y. Lei, B. Yang, Geom-GCN: geometric graph convolutional networks, in: ICLR, OpenReview.net, 2020.
[21] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 1725–1735.
[22] R. Li, S. Wang, F. Zhu, J. Huang, Adaptive graph convolutional neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018.
[23] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 5453–5462.
[24] Y. Rong, W. Huang, T. Xu, J. Huang, DropEdge: towards deep graph convolutional networks on node classification, in: 8th International Conference on Learning Representations, OpenReview.net, 2020.
[25] Y. Wang, T. Derr, Tree decomposed graph neural network, in: CIKM, ACM, 2021, pp. 2040–2049.
[26] J. Zhu, R.A. Rossi, A. Rao, T. Mai, N. Lipka, N.K. Ahmed, D. Koutra, Graph neural networks with heterophily, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 11168–11176.
[27] T. Wang, D. Jin, R. Wang, D. He, Y. Huang, Powerful graph convolutional networks with adaptive propagation mechanism for homophily and heterophily, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 4210–4218.
[28] D. He, C. Liang, H. Liu, M. Wen, P. Jiao, Z. Feng, Block modeling-guided graph convolutional neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 4022–4029.
[29] M.-L. Zhang, B.-B. Zhou, X.-Y. Liu, Partial label learning via feature-aware disambiguation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1335–1344.
[30] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, J. Bailey, Symmetric cross entropy for robust learning with noisy labels, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 322–330.
[31] W. Cong, M. Ramezani, M. Mahdavi, On provable benefits of depth in training graph convolutional networks, Adv. Neural Inf. Process. Syst. 34 (2021).
[32] H. Wang, J. Leskovec, Unifying graph convolutional neural networks and label propagation, arXiv preprint, arXiv:2002.06755, 2020.
[33] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (2008) 93–106.
[34] G. Namata, B. London, L. Getoor, B. Huang, U. Edu, Query-driven active surveying for collective classification, in: 10th International Workshop on Mining and Learning with Graphs, vol. 8, 2012, p. 1.
[35] A. Bojchevski, S. Günnemann, Deep Gaussian embedding of graphs: unsupervised inductive learning via ranking, in: International Conference on Learning Representations, 2018.
[36] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, arXiv preprint, arXiv:1811.05868, 2018.
[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch, NeurIPS-W, 2017.
[38] M. Fey, J.E. Lenssen, Fast graph representation learning with PyTorch geometric, arXiv preprint, arXiv:1903.02428, 2019.

[39] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: International Conference on Machine Learning, 2016, pp. 40–48.
[40] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, in: Proceedings of ICLR, 2015.
[41] M. Liu, H. Gao, S. Ji, Towards deeper graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 338–348.
[42] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, J. Tang, Node similarity preserving graph convolutional networks, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 148–156.
[43] Y. Rong, W. Huang, T. Xu, J. Huang, DropEdge: towards deep graph convolutional networks on node classification, in: 8th International Conference on Learning Representations, OpenReview.net, 2020.
[44] E. Dai, C. Aggarwal, S. Wang, NRGNN: learning a label noise resistant graph neural network on sparsely and noisily labeled graphs, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 227–236.
[45] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, J. Tang, Graph structure learning for robust graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 66–74.